Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7441-93-7012

Interim Report — Map Data Support for AN/SWG–1A(V) Harpoon Ship Command–Launch Control System (HSCLCS)

JAMES H. HAMMACK

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

January 1993

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. Agency Use Only *(Leave blank).*

2. Report Date.
January 1993

3. Report Type and Dates Covered.
Final

4. Title and Subtitle.
Interim Report–Map Data Support for AN/SWG-1A(V) Harpoon Ship Command-Launch Control System

5. Funding Numbers.
Program Element No. OPN
Project No.
Task No.
Accession No. E9133351
Work Unit No. 93512R

6. Author(s).
James A. Hammack

7. Performing Organization Name(s) and Address(es).
Naval Research Laboratory
Mapping, Charting and Geodesy Branch
Stennis Space Center, MS 39529-5004

8. Performing Organization Report Number.
Memorandum Report 7012
NOARL Technical Note 278

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Defense Mapping Agency, Systems Center
8613 Lee Highway
Fairfax, VA 22031-2138

10. Sponsoring/Monitoring Agency Report Number.
Memorandum Report 7012
NOARL Technical Note 278

11. Supplementary Notes.
*This research was performed by the Naval Oceanographic and Atmospheric Research Laboratory (NOARL), recently designated the Naval Research Laboratory. Formerly NOARL Technical Note 278.

12a. Distribution/Availability Statement.
Approved for public release; distribution is unlimited. Naval Research Laboratory, Washington, DC 20375-5320.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

The Harpoon Weapon System (HWS) is an all-weather, over-the-horizon, antiship missile deployed on U.S. Navy vessels and aircraft. Shipborne versions of the HWS are comprised of the Harpoon missile, missile launchers, and the Harpoon Ship Command-Launch Control System (HSCLCS). In order the add the capability to display geographic information on the HSCLCS, the Naval Research Laboratory (NRL) has developed databases and retrieval/display algorithms for both vector shoreline data and gridded landmask data. The shoreline database used is a thinned and compressed version of the Defense Mapping Agency's (DMA) World Vector Shoreline (WVS). The landmask database is a 1-minute grid of land/water values derived from the WVS. Extensive quality control procedures, including paper plots of the entire world and workstation display of all shoreline regions, has ensured that the two databases are in agreement and that the data are correct.

14. Subject Terms.
World Vector Shoreline, Obstructions, Mapping

15. Number of Pages.
33

16. Price Code.

17. Security Classification	18. Security Classification of Report.	19. Security Classification of This Page.	20. Limitation of Abstract.
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

ABSTRACT

The Harpoon Weapon System (HWS) is an all-weather, over-the-horizon, antiship missile deployed on U.S. Navy vessels and aircraft. Shipborne versions of the HWS are comprised of the Harpoon missile, missile launchers, and the Harpoon Ship Command-Launch Control System (HSCLCS). In order to add the capability to display geographic information on the HSCLCS, the Naval Research Laboratory (NRL) has developed databases and retrieval/display algorithms for both vector shoreline data and gridded landmask data. The shoreline database used is a thinned and compressed version of the Defense Mapping Agency's (DMA) World Vector Shoreline (WVS). The landmask database is a 1-minute grid of land/water values derived from the WVS. Extensive quality control procedures, including paper plots of the entire world and workstation display of all shoreline regions, has ensured that the two databases are in agreement and that the data are correct.

ACKNOWLEDGMENTS

The work described in this paper has been sponsored by the Naval Sea Systems Command, Program Element OPN, Program Manager Mr. Lee Minin, NAVSEA 62H. I would like to thank Mr. Danny Clark of the Naval Ship Weapons System Engineering Station and Mr. Jay Paige of McDonnell Douglas Missile Systems Company. Mr. Jerry Landrum, Ms. Danette Coughlan, and Mr. Jonathan Wright of NRL have made significant contributions to this project, and their efforts are appreciated.

The mention of commercial products or the use of company names does not in any way imply endorsement by the U.S. Navy or NRL.

Interim Report-Map Data Support for AN/SWG-1A(V) Harpoon Ship Command-Launch Control System (HSCLCS)

BACKGROUND

The Harpoon Weapon System (HWS) is an all-weather, over-the-horizon, antiship missile designed for use on U.S. Navy ships, submarines, and aircraft. Harpoon is now serving as the Navy's basic antiship missile for Fleet-wide use, being deployed on cruisers, destroyers, frigates, and patrol hydrofoils.

Shipborne versions of the HWS are comprised of the Harpoon missile, missile launchers, and the Harpoon Ship Command-Launch Control System (HSCLCS). The HSCLCS configuration now in use, AN/SWG-1A(V), has a Graphic Display System that can display the designated target and other hostile, friendly, or neutral ships in the area of interest. The HSCLCS will automatically generate and display an optimized flight path to the target. The operator can immediately launch the missile utilizing this flight plan or he can alter this missile flight plan in order to conceal the launch point, fly around other ships or landmasses, avoid defensive systems, or approach the target from multiple directions.

The HSCLCS is being upgraded to provide coastline and landmass display capability. The HSCLCS currently performs the functions of mission planning, launch, and control without displaying landmasses or accounting for them during automatic planning. Effects of land must be manually assessed and the flight plan modified by the operator. In the near term, the HSCLCS is being upgraded to provide landmass and coastline display as visual aids for engagement planning. Future plans call for integration of the map data into the automatic engagement planning algorithms to ensure land avoidance.

The Graphic Display System provides a 9.6" high x 10.7" wide plasma display with 60 pixels/inch resolution. The graphic display is monochrome at range scales of 256, 192, 128, 64, 32, and 16 kiloyards radius, with a viewing area limited to a four inch radius circle. The 32 kiloyard scale is used for most search pattern assessments, with the 16 kiloyard scale selectable if required.

The upgraded HSCLCS will use an Intel 80486 microprocessor, 8 megabytes of high-speed RAM, and 24 megabytes of high-speed, nonvolatile memory. The Naval Research Laboratory (NRL) is developing databases and retrieval algorithms to support display of land areas anywhere in the world under these system constraints. Geographic data will be updated as required by vessel movement. The databases are intended to reside in flash memory until required for display updates. At that time, geographic sections will be moved from the high-speed memory into RAM and then used to update the display.

In order to provide a high-resolution geographic display with sufficient update rates and limited screen clutter, the system will use two compressed databases. Coastline vectors will be provided by thinning and compressing the Defense Mapping Agency's (DMA's) World Vector Shoreline (WVS). Landmask data will be provided by a compressed gridded database of 1' land/water values derived from the WVS data.

COASTLINES

The WVS is a vector database developed and supported by DMA. WVS data are derived primarily by vector processing Digital Landmass Blanking (DLMB) data, with some data in polar regions digitized from paper charts or taken from DMA's Digital Terrain Elevation Data (DTED). DLMB and DTED are gridded data bases with a post spacing of 3 arc-seconds (approximately 100 meters). Data density for the WVS averages approximately 12 data points per nautical mile, or the equivalent of a 1:250,000 scale map. The accuracy requirement for WVS data is that 90% of all identifiable shoreline features be located within 500 m circular error of their true geographic positions with respect to the World Geodetic System (WGS) 84 datum.

COASTLINE THINNING

WVS data were found to have a high percentage of production "noise" and redundant data points. The data were thinned by the Naval Research Laboratory using the Douglas-Peucker algorithm with a thinning threshold of 2 arc-seconds (Landrum, 1989). The 2 arc-second (approximately 60 m) thinning is essentially a noise-removal process and results in a decrease in file size of about 28% with no significant decrease in data quality.

Differences in display of the original DMA WVS data and display of the thinned data set on the Graphic Display System will not exceed 1 pixel width (0.0167 inches) for a 16 kiloyard display. Differences at the 256 kiloyard range scale are on the order of 0.001 inches.

COASTLINE COMPRESSION

After thinning, the data were compressed using the Depner-Hammack compression method described in **A Portable Method for Compression, Storage and Retrieval of High-Resolution Geographic Data Sets** (Hammack, 1990). This compression technique is lossless and allows the storage of a single geographic data point in about 2.5 bytes. The resulting WVS coastline database size is about 18 megabytes.

FILE STRUCTURE

The coastline database file contains two major sections. The first is a bitmap that indicates the 1^{o} cells that contain coastline. The second section contains the actual latitude and longitude coordinates of the coastline segments. The coastline file structure is shown in Figure 1.

Algorithm addressing is used in this compression method to provide a unique record number for every 1^{o} cell on the surface of the Earth. The 1^{o} cells are defined by the latitude and longitude of their lower left (southwest) corner. The record number is computed by:

```
record_number = (latitude degrees + 90) * 360 + (longitude degrees + 180) + 1
```

where **latitude degrees** is the integer portion of the latitude (south is negative) and **longitude degrees** is the integer portion of the longitude (west is negative). This formula yields a record number between 1 and 64,800 for any cell on the surface of the Earth.

The majority of 1^{o} cells on the surface of the Earth do not contain any coastline information. A 64,800-bit bitmap (1 bit per 1^{o} cell) is used to indicate which cells contain coastline. This bitmap is stored in the records 2 through 23. The records corresponding to cells that do not contain coastline are used to store overflow records from other cells. In this way storage is not wasted by empty records.

The use of a 1^{o} bitmap provides a method for quickly determining whether coastline data are available in a cell. After the bitmap is checked to see if any data are available in a 1^{o} cell, the record number is used to directly access the data stored in the second section of the file.

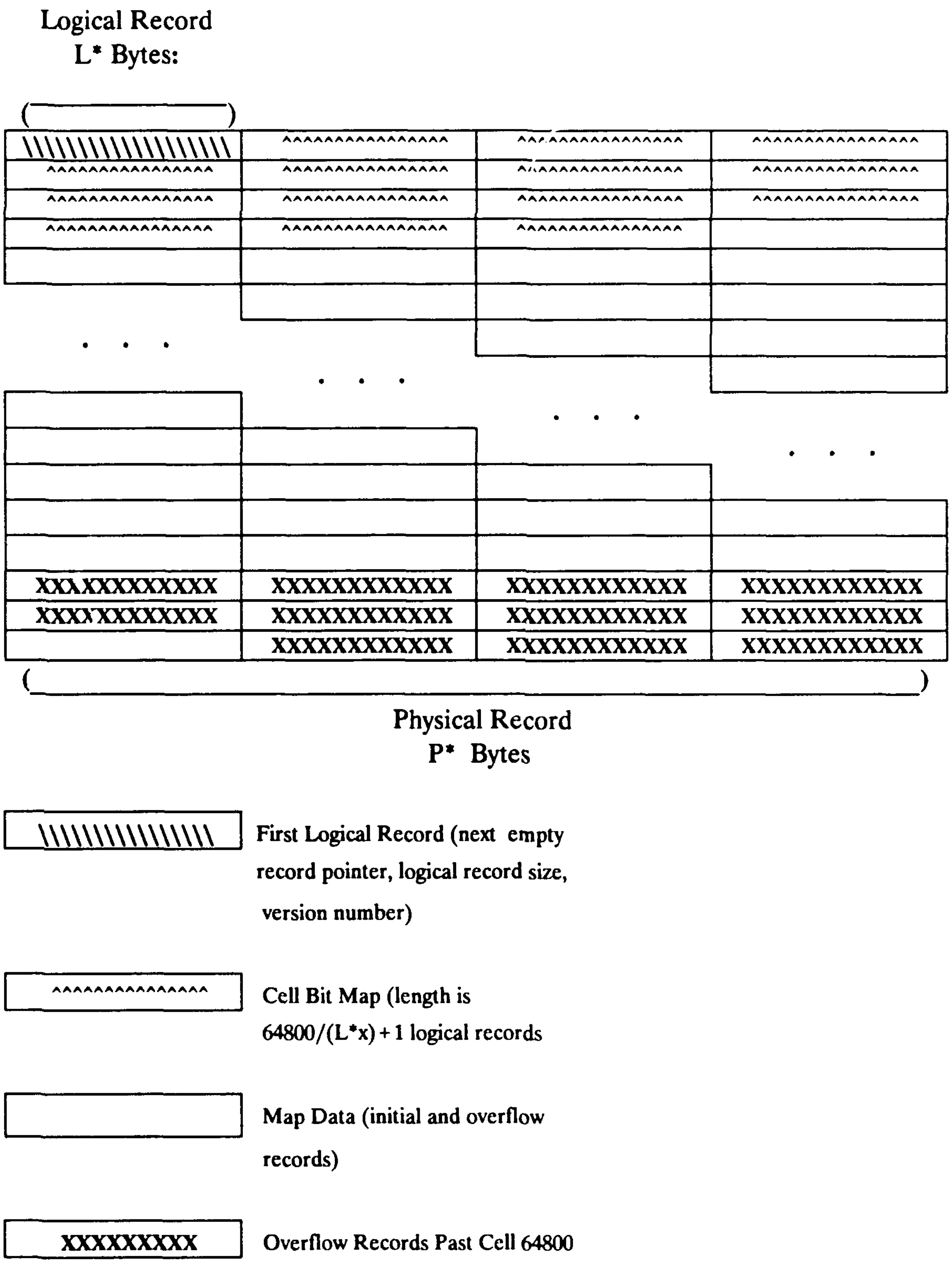

Figure 1. General Direct-Access File Structure.

In the second section of the file, coastline segments are stored as lines that begin and end within a single cell. Coastline segments which would traverse more than one cell are broken at the cell boundaries during the compression process and treated as a series of smaller segments. Each of these segments is contained completely within one cell, with the exception of one overlap point at the cell boundary. This overlap point is required to provide a smooth connection to the bordering cell and eliminate gaps at the cell boundaries.

The computed record number is used as a pointer to the 3072-byte record containing the first of the coastline coordinates for the given 1° cell. Multiple records are chained together to provide sufficient storage space when required. If there are more data than can be stored in a single 3072-byte record, an "overflow pointer" gives the address of the next record in the chain.

Within an individual cell, coastline segments are stored using a **segment header** and **delta records**. Each coastline segment has a single segment header and multiple delta records. The segment header defines three attributes of the line segment: (1) the beginning latitude and longitude of the segment, (2) the number of points in the segment, and (3) a "rank" or identifier for the type of segment. For this application all coastline segments are of the same type, so the "rank" is not used. The structure of the segment header is shown in Figure 2.

The segment header is 6 bytes in length. The first bit of byte 1 contains the continuation flag. If set, this indicates that this record is a continuation of a previous record in the same cell. The remaining seven bits of BYTE 1 contain the most significant seven bits of the segment length (number of points) and the first four bits of BYTE 2 contain the least significant four bits of the segment length.

The degrees portion of the starting latitude and longitude is implicit in the record number and need not be stored. The minutes portion of the starting latitude and longitude is biased by 128, so that negative values may be stored, and stored in BYTES 5 and 6. It is necessary to store negative values in order to accommodate the overlap points needed when a coastline segment crosses a cell boundary. (Negative values would be required if the line segment exits the west or south side of the cell.) The seconds portion of the latitude is split between BYTES 2 and 3 as shown in Figure 1, and the seconds portion of the longitude is stored in a portion of BYTE 3.

SEGMENT HEADER:

1	2	3	4	5	6
			RANK	LATITUDE MINUTES	LONGITUDE MINUTES

BYTE 1 of the SEGMENT HEADER:

Bit:

1	2	3	4	5	6	7	8
Cont. Flag	MOST SIGNIFICANT 7 BITS OF THE SEGMENT LENGTH						

BYTE 2 of the SEGMENT HEADER:

Bit:

1	2	3	4	5	6	7	8
LEAST SIGNIFICANT 4 BITS OF THE SEGMENT LENGTH				MOST SIGNIFICANT 4 BITS OF THE LATITUDE SECONDS			

BYTE 3 of the SEGMENT HEADER:

Bit:

1	2	3	4	5	6	7	8
LEAST SIGNIFICANT 2 BITS OF LATITUDE SECONDS		LONGITUDE SECONDS					

Figure 2. Segment Header Data Structure.

DELTA RECORDS

Following the segment header is a series of delta records. Each delta record consists of two bytes; a latitude delta and a longitude delta. Each delta is the offset in seconds from the previous point. These delta records are also biased by 128 to accommodate negative values.

The last delta value for most line segments will not correspond to the last point in the 1^{o} cell. These last values are normally overlap points, and will also be the first point of a new segment in an adjacent 1^{o} cell.

Details of the compression method and data formats are included in **A Portable Method for Compression, Storage, and Retrieval of High-Resolution Geographic Data Sets** (Depner & Hammack, 1988).

Retrieval and display software has been written in FORTRAN 77 and ANSI C and is included as Appendices A and B, respectively.

LANDMASK

The landmask database consists of a simple 1-minute grid of land and water values derived from the WVS database. The Naval Ocean Systems Center (NOSC) used the WVS database to develop a 3 arc-second grid of country codes (Houghton, 1989). This database was thinned to provide a 1-minute grid of land/water values, then compressed to about 720 kilobytes from the original 49 megabyte NOSC file.

LANDMASK THINNING

The first step in developing the 1' landmask file was to thin the original 3 arc-second data to 1 minute and remove country code information. The NOSC data were run-length coded by country. The NRL thinning algorithm sampled the NOSC database at 1-minute increments and translated country codes into a single "land" code. This thinning resulted in a 30 megabyte file of land and water values, with each bit corresponding to a 1-minute cell. These data were overlayed with WVS data on paper using a CalComp electrostatic color plotter in order to detect any bad data values in the file. The file was then manually edited on a Sun SPARCstation 2 in an X-window environment to correct bad data points. The final data file was then again plotted on the CalComp plotter as a quality control check.

LANDMASK COMPRESSION

After final edit and quality control of the uncompressed data, the database was compressed using a technique that combines bitmaps and run-length coding. Nearly 90% of the 1° cells covering the Earth's surface do not contain shoreline. The 3600 1-minute cells within each of these 1° cells may be represented by a single land/water value, resulting in a significant decrease in storage requirements. One technique for representing these grids is to create bitmaps for larger geographical areas so that many 1-minute cells may be represented by a single value.

The file structure used here utilizes both 10° bitmaps and 1° bitmaps. If a given 10° area is entirely land or water, then this information is obtainable from the bitmap and no further access of the database is necessary. If a 10° square contains both land and water, it is then necessary to retrieve the appropriate 1° bitmap. However, if the 1° square in question is entirely land or water, no further access is required. It is only when a 1° square contains both land and water that access to the run-length coded information is required. Using this method, the majority of database inquiries require two or fewer reads, even if geographic points are accessed randomly.

The logical record size used for the landmask database is 32 bytes. The first four bytes of the first record contain the number of 1' land cells in the database. The fifth byte contains the database revision level. The Harpoon database is identified by revision level 2. Level 1 data are based on the Central Intelligence Agency (CIA) World Data Bank II coastline, and Level 0 is not used. The following bytes provide a textual description of the database; in this case, "NRL 1' WVS Landmask."

The following 81 logical records (2592 bytes) in the file contain the 10° bitmap. Each four-byte integer contains either an address, if both land and water are present in the square, or a flag value indicating that the square is either all land or all water. In keeping with long-standing military tradition, the flag is set to "1 if by land, 2 if by sea." The addresses contained in this section point to a 1° square map record.

Each 1° bitmap contains a two-bit flag for each 1° square within its parent 10° square. This flag indicates whether the 1° square is all water (2), all land (1), or both (0). Immediately following this 1° bitmap is the "initial" run-length coded data record for the first 1° square that contains both land and water. The address of any initial 1° square record can be computed from the number of "0" records preceding it (address = number of preceding "0" records * 32).

If the run-length coded data for an entire 1° square do not fit in one logical record (32 bytes), then the last two bytes of the "initial" record contain a pointer to an "overflow" record. Similarly, overflow records may also contain pointers to other overflow records. All overflow records are stored after the block of "initial" records in the file (see Figure 3).

The 1° run-length coded records were built by dividing the square into 60, 1-minute latitude bands, then placing these latitude bands end-to-end (starting from the southern edge) to produce a 3600 point array. This array is then compressed using run-length coding. Each logical piece (sub-record) of the run-length record may be either one or two bytes in length, depending on the size of the run stored. This sub-record consists of a flag indicating the number of bits used to store the run-length (either 6 or 14 bits), a flag that indicates whether the run is land or water, and the run length. The first flag is 1 bit in length and has a value of 0 if the run length is stored in 6 bits, or a value of 1 if the run length is stored in 14 bits. The second flag is 1 bit in length, and is set to 0 if the run is water or 1 if the run is land. The following 6 or 14 bits contain a count of the number of points in the run. If there are too many data to fit into one logical record (32 bytes), then the last 2 bytes of the record will contain a pointer to the next overflow record. This pointer is stored as an offset from the current position. Overflow records are placed immediately after the initial 1° run-length coded records in the file, but prior to the next 1° bitmap record (see Figure 3).

RETRIEVAL AND DISPLAY SOFTWARE

Retrieval and display software was developed for both the WVS and landmask data files. Retrieval code was written both in ANSI C and in FORTRAN 77, and sample display code was provided for the IBM PC/MS-DOS environment. This software allows rapid access to any 1° cell on the surface of the Earth.

SUMMARY

In order to add the capability to display geographic information on the HSCLCS, NRL has developed databases and retrieval/display algorithms for both vector shoreline data and gridded landmask data. The shoreline data used is a thinned and compressed version of DMA's WVS. The landmask data is a 1-minute grid of land/water values derived from the WVS. Extensive quality control procedures, including paper plots of the entire world and workstation display of all shoreline regions, have ensured that the two databases are in agreement and that the data is correct.

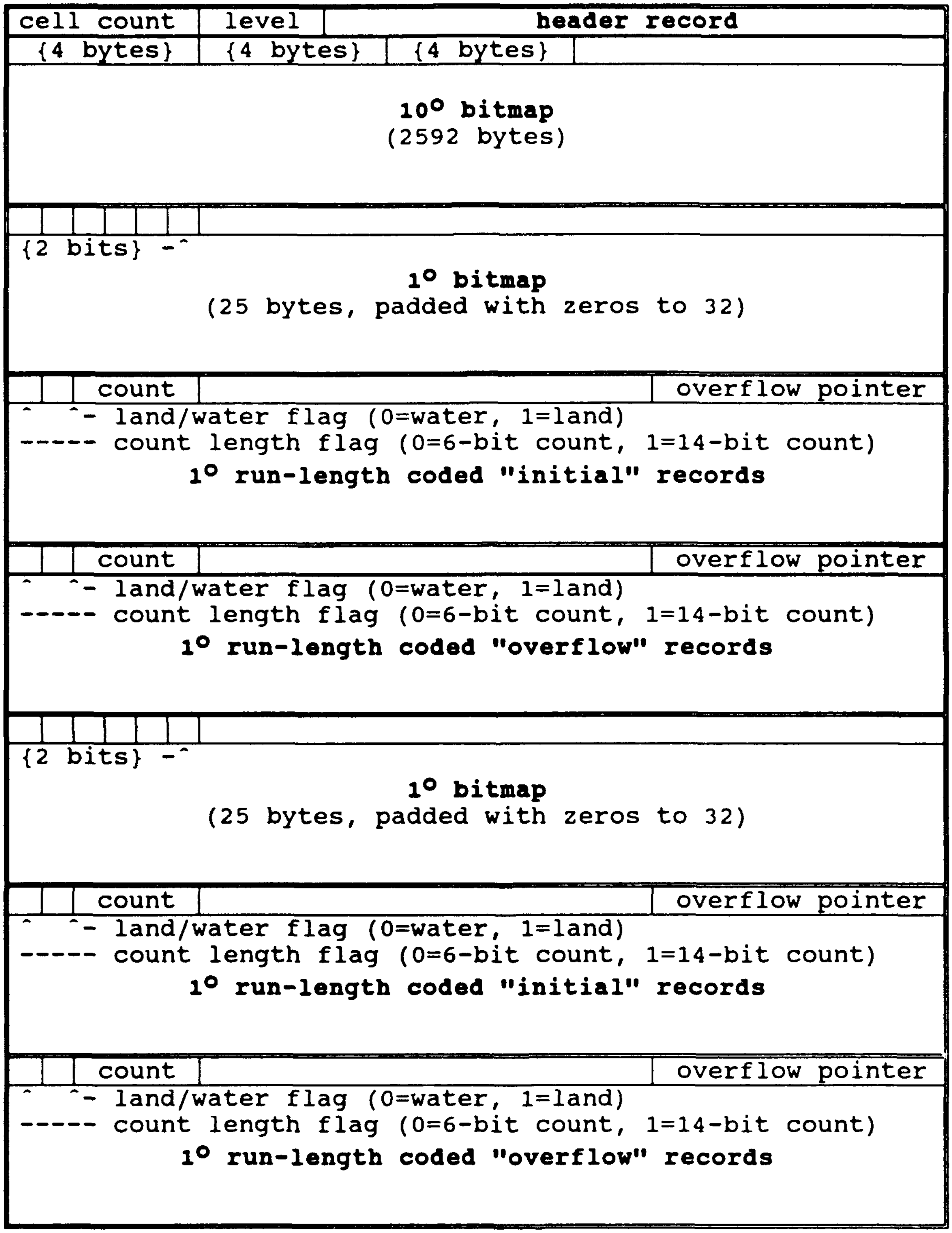

Figure 3. Landmask File Structure.

REFERENCES

Depner, Jan C., and James A. Hammack (1988). A Portable Method for Compression, Storage, and Retrieval of High-Resolution Geographic Data Sets. Naval Oceanographic Office, SSC, MS, NAVOCEANO Technical Note TN 01-88.

Hammack, James A. (1990). A Portable Method for Compression, Storage, and Retrieval of High-Resolution Geographic Data Sets. Proceedings of the Naval Digital Mapping, Charting, & Geodesy Analysis Program Interest Group Meeting.

Houghton, Jack (1989). Description of the NOSC 3 Arc-second Landmask (private communication).

Landrum, Jerry L. (1989). Simplification Methods for the World Vector Shoreline. Naval Ocean Research and Development Activity, SSC, MS, NORDA Report 232.

APPENDIX A

```
C
C     FUNCTION MASK01 : RETURNS A LOGICAL FLAG INDICATING THAT THE
C     ONE MINUTE GRID NEAREST TO THE LATITUDE AND LONGITUDE POSITION
C     PASSED TO THE SUBROUTINE IS EITHER ON LAND OR WATER.  THE POINT
C     MAY BE DEFINED AS DEGREES DECIMAL OR DEGREES AND MINUTES DECIMAL.
C     THE FLAG RETURNED IS SET TO .TRUE. FOR LAND OR .FALSE. FOR WATER.
C     THIS ROUTINE CALLS MASK60 TO GET THE DEGREE SQUARE MASK.
C
C                THIS PROGRAM USES A DATA BASE DESIGNED BY:
C
C            JAN C. DEPNER                        JAMES A. HAMMACK
C            NAVAL OCEANOGRAPHIC OFFICE           NAVAL OCEAN R&D ACTIVITY
C            SYSTEMS TECHNOLOGY BRANCH            CODE 117
C            STENNIS SPACE CENTER, MS             STENNIS SPACE CENTER, MS
C            39522-5001                           39529-
C
C            AUTHOR : JAN C. DEPNER, 04/03/89
C
C     ARGUMENTS :
C
C        LATD     -  (R) LATITUDE DEGREES OF THE POINT (SOUTH NEGATIVE)
C        LATM     -  (R) LATITUDE MINUTES OF THE POINT (UNSIGNED UNLESS
C                    LATD IS 0.0)
C        LOND     -  (R) LONGITUDE DEGREES OF THE POINT (WEST NEGATIVE OR
C                    0 TO 360)
C        LONM     -  (R) LONGITUDE MINUTES OF THE POINT (UNSIGNED UNLESS
C                    LOND IS 0.0)
C        MASK01   -  (L) POINT FLAG RETURNED
C
C     VARIABLES :
C
C        MASK     -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
C                    MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
C                    THE LATITUDE AND LONGITUDE POINT REQUESTED
C        LATP     -  (I) LATITUDE INDEX INTO MASK ARRAY
C        LONP     -  (I) LONGITUDE INDEX INTO MASK ARRAY
C        LAT      -  (R) LATITUDE IN DECIMAL DEGREES
C        LON      -  (R) LONGITUDE IN DECIMAL DEGREES
C
      LOGICAL FUNCTION MASK01(LATD,LATM,LOND,LONM)
      IMPLICIT INTEGER (A-Z)
      REAL LATD,LATM,LOND,LONM,LAT,LON
      CHARACTER*1 MASK(0:59,0:59),ALFLAG,FLAG60
      DATA PLAT /-1/ , PLON /-1/ , FLAG60 /'M'/
C
C     CONVERT LAT AND LON TO DEGREES DECIMAL, ADJUST TO NEAREST MINUTE
C
      LAT = LATD+(LATM/60.0)*SIGN(1.0,LATD)+90.0
      LON = LOND+(LONM/60.0)*SIGN(1.0,LOND)
      IF (LON.LT.0.0) LON = LON + 360.0
      LAT = LAT+.008333333333333
      LON = LON+.008333333333333
C
C     IF THIS IS IN THE SAME ONE DEGREE SQUARE AS THE LAST ACCESS,
C     DO NOT READ A NEW MASK
C
      DLAT = INT(LAT)
      DLON = INT(LON)
C
C     GET THE ONE DEGREE MASK
C
      IF (DLON.NE.PLON.OR.DLAT.NE.PLAT) CALL MASK60(MASK,LAT-90.0,LON,
     &FLAG60)
```

```
C
C     IF THE ALL LAND OR WATER FLAG IS NOT SET, GET THE SINGLE POINT
C     FROM THE MASK
C
      IF (FLAG60.EQ.' ') THEN
       LATP = AMOD(LAT,1.0)*60.0
       LONP = AMOD(LON,1.0)*60.0
       ALFLAG = MASK(LONP,LATP)
      ELSE
       ALFLAG = FLAG60
      END IF
C
C     SET THE LOGICAL VALUE BASED ON THE SETTING OF ALFLAG
C
      IF (ALFLAG.EQ.'L') THEN
       MASK01 = .TRUE.
      ELSE
       MASK01 = .FALSE.
      END IF
C
C     SAVE THE PREVIOUS LAT AND LON DEGREES
C
      PLAT = DLAT
      PLON = DLON
      RETURN
      END
C
C     SUBROUTINE MASK60 : GIVEN A REAL LATITUDE AND LONGITUDE IN DEGREES
C     (WEST LONGITUDE NEGATIVE OR 0 TO 360 LONGITUDE AND SOUTH LATITUDE
C     NEGATIVE) THIS SUBROUTINE RETURNS A CHARACTER ARRAY CONTAINING A
C     ONE MINUTE LAND MASK FOR THE ONE DEGREE CELL THAT THE LATITUDE
C     AND LONGITUDE POINT IS IN.  THE CHARACTER ARRAY CONTAINS 3600
C     CHARACTERS AND IS SET UP AS 'CHARACTER*1 MASK(0:59,0:59)'.  EACH
C     CHARACTER IS SET TO EITHER 'L' OR 'W'.  IF ALL THE POINTS ARE
C     LAND OR ALL ARE WATER, THE 1 CHARACTER FLAG 'ALFLAG' WILL BE SET
C     TO THE CORRESPONDING LETTER.  IF BOTH LAND AND WATER POINTS ARE
C     PRESENT, 'ALFLAG' WILL BE SET TO ' '.  THE MASK IS SET UP WITH
C     POINT 'MASK(0,0)' BEING THE SOUTHWEST CORNER OF THE CELL AND POINT
C     'MASK(59,59)' THE NORTHEAST CORNER.  THE FIRST SUBSCRIPT REPRESENTS
C     THE LONGITUDE MINUTES, THE SECOND IS LATITUDE MINUTES (WITH THE
C     COORDINATE SYSTEM SET UP AS 0 - 360 LONGITUDE AND 0 - 180 LATITUDE).
C
C                THIS PROGRAM USES A DATA BASE DESIGNED BY:
C
C            JAN C. DEPNER                        JAMES A. HAMMACK
C            NAVAL OCEANOGRAPHIC OFFICE           NAVAL OCEAN R&D ACTIVITY
C            SYSTEMS TECHNOLOGY BRANCH            CODE 117
C            STENNIS SPACE CENTER, MS             STENNIS SPACE CENTER, MS
C            39522-5001                           39529-
C
C            AUTHOR : JAN C. DEPNER, 04/03/89
C
      SUBROUTINE MASK60(MASK,LAT,LON,ALFLAG)
      IMPLICIT INTEGER (A-Z)
      REAL LAT,LON
      PARAMETER (PHYREC=1024)
      CHARACTER*1 MASK(0:59,0:59),ALFLAG
      COMMON /ADDRS/ LPERP,REC00,REC01
      DATA FIRST /1/ , PLAT /-1/ , PLON /-1/
C
C     IF FIRST TIME THROUGH, OPEN FILE AND SET LPERP
C
      IF (FIRST.EQ.1) THEN
C
```

```
C          RECORD LENGTH FOR FILE IS PHYREC BYTES (CHECK YOUR MACHINE FOR
C          RECORD LENGTH IN BYTES OR WORDS)
C
         OPEN(10,FILE='data/landmask.wvs',FORM='UNFORMATTED',
     &    ACCESS='DIRECT',STATUS='OLD',RECL=PHYREC)
          LPERP = PHYREC/32
          FIRST = 0
       END IF
C
C      IF LAT >= 90.0 SET TO ALL WATER AND RETURN
C
       IF (LAT.GE.90.0) THEN
        ALFLAG = 'W'
        CALL ALLMSK(ALFLAG,MASK)
        RETURN
       END IF
C
C      CALCULATE SOUTHWEST CORNER OF CELL
C
       CLAT = INT(LAT+90.0)
       IF (LON.LT.0) THEN
          CLON = INT(LON+360.0)
       ELSE
          CLON = INT(LON)
       END IF
C
C      IF THE LAT AND LON ARE NOT IN THE SAME ONE DEGREE SQUARE AS THE
C      PREVIOUS LAT AND LON, OR ALFLAG = 'M' (FIRST CALL FROM MASK01),
C      PROCESS THE CELL
C
       IF (CLON.NE.PLON.OR.CLAT.NE.PLAT.OR.ALFLAG.EQ.'M') THEN
          ALFLAG = ' '
C
C         READ THE TEN DEGREE SQUARE MAP AND CHECK FOR ALL LAND OR WATER
C
          CALL RDTEN(CLAT,CLON,ALFLAG)
          IF (ALFLAG.NE.' ')THEN
             CALL ALLMSK(ALFLAG,MASK)
          ELSE
C
C            READ THE ONE DEGREE CELL MAP AND CHECK FOR ALL LAND OR WATER
C
             CALL RDONE(CLAT,CLON,ALFLAG)
             IF (ALFLAG.NE.' ')THEN
              CALL ALLMSK(ALFLAG,MASK)
             ELSE
C
C               READ THE RUN LENGTH DATA FOR THE ONE DEGREE CELL AND BUILD
C               THE MASK
C
                CALL BLDMSK(MASK)
             END IF
          END IF
C
C         SAVE THE LAT AND LON DEGREES
C
          PLAT = CLAT
          PLON = CLON
       END IF
       RETURN
       END
C
C      SUBROUTINE RDTEN : READS THE TEN DEGREE SQUARE MAP AT THE
C      BEGINNING OF THE DATA BASE.  RETURNS A FLAG INDICATING IF THE
```

```
C     ENTIRE SQUARE WAS EITHER LAND OR WATER.  IF NOT, THE VARIABLE
C     'REC01' IS SET TO THE LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
C     THE FLAG RETURNED IS SET TO 'L' FOR LAND OR 'W' FOR WATER.
C
C            AUTHOR : JAN C. DEPNER, 04/03/89
C
C     ARGUMENTS :
C
C        LAT       -  (R) LATITUDE DEGREES OF THE POINT (0 - 180)
C        LON       -  (R) LONGITUDE DEGREES OF THE POINT (0 - 360)
C        TENFLG    -  (C) ALL LAND/WATER FLAG RETURNED ('L' OR 'W')
C
C     VARIABLES :
C
C        PHYREC    -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
C                     RECORD IN THE DATA BASE.
C        IREC10    -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
C                     DATA BASE.
C        CTYPE     -  (C) ARRAY CONTAINING THE LAND AND WATER FLAGS.
C        LPERP     -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
C        REC00     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
C                     (NOT USED IN THIS ROUTINE).
C        REC01     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
C        PADDR     -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
C
      SUBROUTINE RDTEN(LAT,LON,TENFLG)
      IMPLICIT INTEGER (A-Z)
      PARAMETER (PHYREC=1024)
      CHARACTER*1 IREC10(0:PHYREC-1),TENFLG,CTYPE(2)
      COMMON /ADDRS/ LPERP,REC00,REC01
      DATA PADDR /-1/ , CTYPE /'L','W'/ , PWORD /-1/
C
C     COMPUTE THE WORD (4 BYTE) POSITION WITHIN THE 648 WORD TEN
C     DEGREE SQUARE MAP, MAP STARTS IN THE 2ND LOGICAL RECORD
C
      WORD = (LAT/10)*36+LON/10
C
C     IF THE WORD POSITION HAS CHANGED GET THE NEW ONE DEGREE CELL
C     LOGICAL ADDRESS
C
      IF (WORD.NE.PWORD) THEN
C
C        COMPUTE THE LOGICAL AND PHYSICAL ADDRESS OF THE ONE DEGREE
C        CELL POINTER WITHIN THE TEN DEGREE MAP
C
         LREC = (WORD*4)/32+1
         ADDR = INT(LREC/LPERP+1)
C
C        IF THE PHYSICAL ADDRESS HAS CHANGED, READ A NEW RECORD
C
         IF (ADDR.NE.PADDR) THEN
            READ(10,REC=ADDR)IREC10
            PADDR = ADDR
         END IF
C
C        COMPUTE THE BYTE POSITION OF THE ONE DEGREE CELL LOGICAL
C        ADDRESS WITHIN THE PHYSICAL RECORD
C
         PNT = MOD((WORD+8)*4,PHYREC)
C
C        BUILD THE LOGICAL ADDRESS FROM THE 4 BYTES AT 'PNT'
C
         REC01 = ICHAR(IREC10(PNT))*16777216+ICHAR(IREC10(PNT+1))*65536+
     &   ICHAR(IREC10(PNT+2))*256+ICHAR(IREC10(PNT+3))
```

```
            PWORD = WORD
         END IF
C
C        IF THE TEN DEGREE SQUARE IS ALL LAND OR WATER, SET THE FLAG
C
         IF (REC01.LT.3) THEN
            TENFLG = CTYPE(REC01)
         ELSE
            TENFLG = ' '
         END IF
         RETURN
         END
C
C        SUBROUTINE RDONE : READS THE ONE DEGREE CELL MAP.  RETURNS A
C        FLAG THAT INDICATES IF THE ENTIRE SQUARE IS EITHER LAND OR
C        WATER.  IF NOT, THE VARIABLE 'REC00' IS SET TO THE LOGICAL
C        ADDRESS OF THE ONE DEGREE DATA RECORD.  THE FLAG RETURNED IS
C        SET TO 'L' FOR LAND OR 'W' FOR WATER.
C
C               AUTHOR : JAN C. DEPNER, 04/03/89
C
C        ARGUMENTS :
C
C           LAT      -  (R) LATITUDE DEGREES OF THE POINT (0 - 180)
C           LON      -  (R) LONGITUDE DEGREES OF THE POINT (0 - 360)
C           ONEFLG   -  (C) ALL LAND/WATER FLAG RETURNED ('L' OR 'W')
C
C        VARIABLES :
C
C           PHYREC   -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
C                       RECORD IN THE DATA BASE.
C           CELMSK   -  (I) ARRAY CONTAINING THE LOGICAL ADDRESSES OF
C                       THE ONE DEGREE CELL DATA RECORDS.
C           IREC01   -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
C                       DATA BASE.
C           CTYPE    -  (C) ARRAY CONTAINING THE LAND AND WATER FLAGS.
C           LPERP    -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
C           REC00    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
C                       (NOT USED IN THIS ROUTINE).
C           REC01    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
C           PADDR    -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
C
         SUBROUTINE RDONE(LAT,LON,ONEFLG)
         IMPLICIT INTEGER (A-Z)
         PARAMETER (PHYREC=1024)
         DIMENSION CELMSK(0:9,0:9)
         CHARACTER*1 IREC01(0:PHYREC-1),ONEFLG,CTYPE(2)
         COMMON /ADDRS/ LPERP,REC00,REC01
         DATA PADDR /-1/ , PREC01 /-1/ , CTYPE /'L','W'/
C
C        COMPUTE THE CELL POSITION WITHIN THE PHYSICAL RECORD AND THE
C        PHYSICAL RECORD ADDRESS
C
         CELL = MOD(REC01,LPERP)*32
         ADDR = INT(REC01/LPERP+1)
C
C        IF LOGICAL RECORD HAS CHANGED, BUILD A NEW CELL MASK
C
         IF (REC01.NE.PREC01) THEN
C
C           IF THE ADDRESS HAS CHANGED, READ A NEW RECORD
C
            IF (ADDR.NE.PADDR) THEN
               READ(10,REC=ADDR)IREC01
```

```
            PADDR = ADDR
         END IF
C
C        BUILD THE CELL MASK FROM THE 2 BIT RECORDS WITHIN THE LOGICAL
C        RECORD; 00 - BOTH LAND AND WATER, 01 - LAND, 10 - WATER.  IF
C        THE 2 BIT RECORD IS 00, SET THE CELL MASK VALUE TO THE LOGICAL
C        ADDRESS OF THE ONE DEGREE DATA RECORD
C
         SUM = 0
         DO 20 I = 0,9
            DO 10 J = 0,9
C
C              COMPUTE THE BIT POSITION WITHIN THE PHYSICAL RECORD, THE
C              BYTE POSITION WITHIN THE PHYSICAL RECORD, AND THE BIT
C              POSITION WITHIN THE BYTE
C
               BPOS = (I*10+J)*2
               BYT = BPOS/8
               BIT = MOD(BPOS,8)
               SET = MOD(ICHAR(IREC01(CELL+BYT))/2**BIT,4)
               IF (SET.EQ.0) THEN
                  SUM = SUM+1
                  CELMSK(J,I) = REC01+SUM
               ELSE
                  CELMSK(J,I) = SET
               END IF
   10       CONTINUE
   20    CONTINUE
         PREC01 = REC01
      END IF
C
C     GET THE ADDRESS, OR GET THE FLAG IF ALL LAND OR WATER
C
      LONP = MOD(LON,10)
      LATP = MOD(LAT,10)
      IF (CELMSK(LONP,LATP).GT.2) THEN
         ONEFLG = ' '
         REC00 = CELMSK(LONP,LATP)
      ELSE
         ONEFLG = CTYPE(CELMSK(LONP,LATP))
      END IF
      RETURN
      END
C
C     SUBROUTINE ALLMSK : SET THE ENTIRE MASK TO EITHER 'L' OR 'W'.
C
C            AUTHOR : JAN C. DEPNER, 04/03/89
C
C     ARGUMENTS :
C
C        MASK     -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
C                    MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
C                    THE LATITUDE AND LONGITUDE POINT REQUESTED
C        ALFLAG   -  (C) ALL LAND/WATER FLAG ('L' OR 'W')
C
      SUBROUTINE ALLMSK(ALFLAG,MASK)
      IMPLICIT INTEGER (A-Z)
      CHARACTER*1 MASK(0:59,0:59),ALFLAG
      DO 20 I = 0,59
         DO 10 J = 0,59
            MASK(J,I) = ALFLAG
   10    CONTINUE
   20 CONTINUE
      RETURN
```

```
      END
C
C     SUBROUTINE BLDMSK : BUILDS THE MASK FROM THE ONE DEGREE DATA
C     RECORDS IN THE DATA BASE.
C
C            AUTHOR : JAN C. DEPNER, 04/03/89
C
C     ARGUMENTS :
C
C        MASK     -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
C                    MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
C                    THE LATITUDE AND LONGITUDE POINT REQUESTED
C
C     VARIABLES :
C
C        PHYREC   -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
C                    RECORD IN THE DATA BASE.
C        IRECOO   -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
C                    DATA BASE.
C        CTYPE    -  (C) ARRAY CONTAINING THE LAND AND WATER FLAGS.
C        LPERP    -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
C        RECOO    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
C                    (NOT USED IN THIS ROUTINE).
C        RECO1    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
C        PADDR    -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
C
      SUBROUTINE BLDMSK(MASK)
      IMPLICIT INTEGER (A-Z)
      PARAMETER (PHYREC=1024)
      CHARACTER*1 IRECOO(0:PHYREC-1),MASK(0:59,0:59),CTYPE(0:1)
      COMMON /ADDRS/ LPERP,RECOO,RECO1
      DATA PADDR /-1/ , CTYPE /'W','L'/
      START = 0
C
C     COMPUTE THE FIRST BYTE POSITION WITHIN THE PHYSICAL RECORD, AND THE
C     PHYSICAL RECORD ADDRESS
C
      POS = MOD(RECOO,LPERP)*32
      ADDR = INT(RECOO/LPERP+1)
C
C     IF THE ADDRESS HAS CHANGED, READ A NEW RECORD
C
      IF (ADDR.NE.PADDR) THEN
         READ(10,REC=ADDR)IRECOO
         PADDR = ADDR
      END IF
C
C     MAIN LOOP, GETS TYPE AND COUNT FROM BYTES.  FIRST TWO BITS ARE TYPE
C     FLAGS; 00 - 1 BYTE WATER, 01 - 1 BYTE LAND, 10 - 2 BYTE WATER,
C     11 - 2 BYTE LAND.  NEXT 6 OR 14 BITS ARE COUNT
C

   10 MBYTE = ICHAR(IRECOO(POS))
      IF (MBYTE.NE.0) THEN
C
C        IF THIS IS A TWO BYTE RECORD, SET DFLAG
C
         IF (MBYTE.GE.128) THEN
            MBYTE = MBYTE-128
            DFLAG = 1
         ELSE
            DFLAG = 0
         END IF
C
```

```
C         GET THE TYPE AND COUNT
C
          TYPE = MBYTE/64
          COUNT = MBYTE-TYPE*64
C
C         IF THIS IS A TWO BYTE RECORD, GET THE REST OF THE COUNT FROM
C         THE NEXT BYTE
C
          IF (DFLAG.EQ.1) THEN
             CALL MOVPOS(POS,IREC00,PADDR)
             COUNT = COUNT*256+ICHAR(IREC00(POS))
          END IF
C
C         LOOP THROUGH THE MASK AND SET THE CHARACTERS
C
          FINISH = START+COUNT-1
          DO 20 J = START,FINISH
             LATP = J/60
             LONP = MOD(J,60)
             MASK(LONP,LATP) = CTYPE(TYPE)
   20     CONTINUE
C
C         MOVE THE BYTE POSITION AND GO TO TOP OF LOOP
C
          CALL MOVPOS(POS,IREC00,PADDR)
          START = FINISH+1
          GO TO 10
       END IF
C
C      FLIP THE TYPE FLAG AND SET THE REMAINDER OF THE MASK
C
   25  TYPE = 1-TYPE
       DO 30 J = START,3599
          LATP = J/60
          LONP = MOD(J,60)
          MASK(LONP,LATP) = CTYPE(TYPE)
   30  CONTINUE
       PTYPE = TYPE
       RETURN
       END
C
C      SUBROUTINE MOVPOS : INCREMENTS THE BYTE POSITION AND CHANGES
C      TO THE NEXT OVERFLOW RECORD IN THE CHAIN IF NECCESSARY.
C
C              AUTHOR . JAN C. DEPNER, 04/03/89
C
C      ARGUMENTS :
C
C         POS       -  (I) BYTE POSITION WITHIN THE PHYSICAL RECORD.
C         IREC00    -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
C                      DATA BASE.
C         PADDR     -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
C
C      VARIABLES :
C
C         PHYREC    -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
C                      RECORD IN THE DATA BASE.
C         LPERP     -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
C         REC00     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
C                      (NOT USED IN THIS ROUTINE).
C         REC01     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
C
       SUBROUTINE MOVPOS(POS,IREC00,PADDR)
       IMPLICIT INTEGER (A-Z)
```

```
      PARAMETER (PHYREC=1024)
      CHARACTER*1 IREC00(0:PHYREC-1)
      COMMON /ADDRS/ LPERP,REC00,REC01
      POS = POS+1
C
C     IF THIS IS THE END OF THE LOGICAL RECORD, GET THE OVERFLOW POINTER
C     FROM THE LAST 2 BYTES OF THE RECORD
C
      IF (MOD(POS+2,32).EQ.0) THEN
         LREC = ICHAR(IREC00(POS))*256+ICHAR(IREC00(POS+1))+REC01
C
C        IF LREC = REC01, THERE IS NO OVERFLOW POINTER AND THIS IS THE
C        END OF THE CHAIN
C
         IF (LREC.NE.REC01) THEN
C
C           COMPUTE THE PHYSICAL ADDRESS, IF IT HAS CHANGED, READ A NEW
C           RECORD
C
            ADDR = LREC/LPERP+1
            IF (ADDR.NE.PADDR) THEN
               READ(10,REC=ADDR)IREC00
               PADDR =ADDR
            END IF
C
C           COMPUTE THE NEW BYTE POSITION WITHIN THE PHYSICAL RECORD
C
            POS = MOD(LREC,LPERP)*32
         END IF
      END IF
      RETURN
      END
```

APPENDIX B

```
#include <math.h>
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <stdio.h>
#include <io.h>

#define PHYREC 1024
#define LPERP PHYREC/32
#define FILENAME "landmask.dat"

#define SIGN_OF(x) ((x)<0.0 ? -1 : 1)

/*
     FUNCTION MASK01 : RETURNS A LOGICAL FLAG INDICATING THAT THE
     ONE MINUTE GRID NEAREST TO THE LATITUDE AND LONGITUDE POSITION
     PASSED TO THE SUBROUTINE IS EITHER ON LAND OR WATER.  THE POINT
     MAY BE DEFINED AS DEGREES DECIMAL OR DEGREES AND MINUTES DECIMAL.
     THE FLAG RETURNED IS SET TO .TRUE. FOR LAND OR .FALSE. FOR WATER.
     THIS ROUTINE CALLS MASK60 TO GET THE DEGREE SQUARE MASK.

            THIS PROGRAM USES A DATA BASE DESIGNED BY:

          JAN C. DEPNER                      JAMES A. HAMMACK
          NAVAL OCEANOGRAPHIC OFFICE         NAVAL OCEAN R&D ACTIVITY
          SYSTEMS TECHNOLOGY BRANCH          CODE 117
          STENNIS SPACE CENTER, MS           STENNIS SPACE CENTER, MS
          39522-5001                         39529-

          AUTHOR : JAN C. DEPNER, 04/03/89

     ARGUMENTS :

      latd      -  (F) LATITUDE DEGREES OF THE POINT (SOUTH NEGATIVE)
      latm      -  (F) LATITUDE MINUTES OF THE POINT (UNSIGNED UNLESS
                 LATD IS 0.0)
      lond      -  (F) LONGITUDE DEGREES OF THE POINT (WEST NEGATIVE OR
                 0 TO 360)
      lonm      -  (F) LONGITUDE MINUTES OF THE POINT (UNSIGNED UNLESS
                 LOND IS 0.0)

     VARIABLES :

      mask      -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
                 MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
                 THE LATITUDE AND LONGITUDE POINT REQUESTED
      latp      -  (I) LATITUDE INDEX INTO MASK ARRAY
      lonp      -  (I) LONGITUDE INDEX INTO MASK ARRAY
      lat       -  (F) LATITUDE IN DECIMAL DEGREES
      lon       -  (F) LONGITUDE IN DECIMAL DEGREES
*/

char mask01(latd,latm,lond,lonm)
float latd, latm, lond, lonm;
{
      static unsigned char mask[3600];
      static char alflag, flag60 = 127;
      float lat, lon;
      double dummy;
      int dlat, dlon, latp, lonp;
      static int plat = -1, plon = -1;

      void mask60();
```

```
/*
      CONVERT LAT AND LON TO DEGREES DECIMAL, ADJUST TO NEAREST MINUTE
*/
      lat = latd + (latm/60.0) * SIGN_OF(latd) + 90.0;
      lon = lond + (lonm/60.0) * SIGN_OF(lond);
      if (lon<0.0) lon += 360.0;
      lat += .00833333333333;
      lon += .00833333333333;
/*
        IF THIS IS IN THE SAME ONE DEGREE SQUARE AS THE LAST ACCESS,
        DO NOT READ A NEW MASK
*/
      dlat = (int)lat;
      dlon = (int)lon;
      if (dlon!=plon || dlat!=plat) {
/*
         GET THE ONE DEGREE MASK
*/
         mask60(mask,lat-90.0,lon,&flag60);
      }
/*
      IF THE ALL LAND OR WATER FLAG IS NOT SET, GET THE SINGLE POINT
      FROM THE MASK
*/
      if (flag60<0) {
         latp = modf((double)lat,&dummy)*60.0;
         lonp = modf((double)lon,&dummy)*60.0;
         alflag = mask[(int)latp*60+(int)lonp];
      }
      else {
         alflag = flag60;
      }
/*
      SAVE THE PREVIOUS LAT AND LON DEGREES
*/
      plat = dlat;
      plon = dlon;
/*
      SET THE LOGICAL VALUE BASED ON THE SETTING OF ALFLAG
*/
      if (alflag) {
         return (1);
      }
      else {
         return (0);
      }
}

/*
      FUNCTION MASK60 : GIVEN A REAL LATITUDE AND LONGITUDE IN DEGREES
      (WEST LONGITUDE NEGATIVE OR 0 TO 360 LONGITUDE AND SOUTH LATITUDE
      NEGATIVE) THIS SUBROUTINE RETURNS A CHARACTER ARRAY CONTAINING A
      ONE MINUTE LAND MASK FOR THE ONE DEGREE CELL THAT THE LATITUDE
      AND LONGITUDE POINT IS IN.  THE CHARACTER ARRAY CONTAINS 3600
      CHARACTERS AND IS SET UP AS 'UNSIGNED CHAR[60][60]'.  EACH
      CHARACTER IS SET TO EITHER '1' OR '0'.  IF ALL THE POINTS ARE
      LAND OR ALL ARE WATER, THE 1 CHARACTER FLAG 'ALFLAG' WILL BE SET
      TO THE CORRESPONDING VALUE.  IF BOTH LAND AND WATER POINTS ARE
      PRESENT, 'ALFLAG' WILL BE SET TO '-1'.  THE MASK IS SET UP WITH
      POINT 'MASK[0][0]' BEING THE SOUTHWEST CORNER OF THE CELL AND POINT
      'MASK[59][59]' THE NORTHEAST CORNER.  THE FIRST SUBSCRIPT REPRESENTS
      THE LONGITUDE MINUTES, THE SECOND IS LATITUDE MINUTES (WITH THE
      COORDINATE SYSTEM SET UP AS 0 - 360 LONGITUDE AND 0 - 180 LATITUDE).
```

```
          THIS PROGRAM USES A DATA BASE DESIGNED BY:

        JAN C. DEPNER                        JAMES A. HAMMACK
        NAVAL OCEANOGRAPHIC OFFICE           NAVAL OCEAN R&D ACTIVITY
        SYSTEMS TECHNOLOGY BRANCH            CODE 117
        STENNIS S6ACE CENTER, MS             STENNIS SPACE CENTER, MS
        39522-5001                           39529-

        AUTHOR : JAN C. DEPNER, 04/03/89
    ARGUMENTS :

     mask     - (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
                MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
                THE LATITUDE AND LONGITUDE POINT REQUESTED
     lat      - (F) LATITUDE DEGREES OF THE POINT (0 - 180)
     lon      - (F) LONGITUDE DEGREES OF THE POINT (0 - 360)
     alflag   - (C) ALL LAND/WATER FLAG RETURNED

    VARIABLES :

     PHYREC   - (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
                RECORD IN THE DATA BASE.
     irec10   - (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
                DATA BASE.
     LPERP    - (I) LOGICAL RECORDS PER PHYSICAL RECORD.
     rec00    - (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
               (NOT USED IN THIS ROUTINE).
     rec01    - (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
     paddr    - (I) PREVIOUS PHYSICAL RECORD ADDRESS.
*/

void mask60(mask,lat,lon,alflag)
char mask[], *alflag;
float lat, lon;
{

      static long rec00, rec01;
      static int first = 1, plat = -1, plon = -1, lunfil;
      int clat, clon;

      void rdten(), rdone(), allmsk(), bldmsk();

/*
        IF FIRST TIME THROUGH, OPEN FILE AND SET LPERP
*/
      if (first) {
         lunfil = open(FILENAME,O_BINARY|O_RDONLY);
         first = 0;
      }
/*
      IF LAT >= 90 SET TO ALL WATER AND RETURN
*/
      if (lat >= 90.0) {
         *alflag = 0;
         allmsk(*alflag,mask);
         return;
      }
/*
        CALCULATE SOUTHWEST CORNER OF CELL
*/
      clat = (int)(lat+90.0);
      if (lon<0.0) {
         clon = (int)(lon+360.0);
      }
```

```
      else {
         clon = ((int)lon)%360;
      }
/*
        IF THE LAT AND LON ARE NOT IN THE SAME ONE DEGREE SQUARE AS THE
        PREVIOUS LAT AND LON, OR ALFLAG = 127 (FIRST CALL FROM MASK01),
        PROCESS THE CELL
*/
      if (clon!=plon || clat!=plat || *alflag==127) {
         *alflag = ' ';
/*
         READ THE TEN DEGREE SQUARE MAP AND CHECK FOR ALL LAND OR WATER
*/
         rdten(lunfil,&rec01,clat,clon,alflag);
         if (*alflag>=0) {
            allmsk(*alflag,mask);
         }
         else {
/*
            READ THE ONE DEGREE CELL MAP AND CHECK FOR ALL LAND OR WATER
*/
            rdone(lunfil,rec01,&rec00,clat,clon,alflag);
            if (*alflag>=0) {
             allmsk(*alflag,mask);
            }
            else {
/*
               READ THE RUN LENGTH DATA FOR THE ONE DEGREE CELL AND BUILD
               THE MASK
*/
             bldmsk(lunfil,rec00,rec01,mask);
            }
         }
/*
         SAVE THE LAT AND LON DEGREES
*/
         plat = clat;
         plon = clon;
      }
}

/*
      FUNCTION RDTEN : READS THE TEN DEGREE SQUARE MAP AT THE
      BEGINNING OF THE DATA BASE.  RETURNS A FLAG INDICATING IF THE
      ENTIRE SQUARE WAS EITHER LAND OR WATER.  IF NOT, THE VARIABLE
      'REC01' IS SET TO THE LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
      THE FLAG RETURNED IS SET TO '1' FOR LAND OR '0' FOR WATER.

           AUTHOR : JAN C. DEPNER, 04/03/89

      ARGUMENTS :

       lunfil   -  (I) INPUT FILE HANDLE
       rec01    -  (L) RECORD NUMBER OF ONE DEGREE CELL MAP
       lat      -  (F) LATITUDE DEGREES OF THE POINT (0 - 180)
       lon      -  (F) LONGITUDE DEGREES OF THE POINT (0 - 360)
       tenflg   -  (C) ALL LAND/WATER FLAG RETURNED

      VARIABLES :

       PHYREC   -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
                  RECORD IN THE DATA BASE.
       irec10   -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
                  DATA BASE.
```

```
       LPERP     -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
       rec00     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
                  (NOT USED IN THIS ROUTINE).
       rec01     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
       paddr     -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
*/

void rdten(lunfil,rec01,lat,lon,tenflg)
int lunfil, lat, lon;
long *rec01;
char *tenflg;
{

      static long paddr = -1, pword = -1;
      static unsigned char irec10[PHYREC];
      long addr, word, lrec, pnt, lstat;

/*
        COMPUTE THE WORD (4 BYTE) POSITION WITHIN THE 648 WORD TEN
        DEGREE SQUARE MAP, MAP STARTS IN THE 2ND LOGICAL RECORD
*/
      word = (lat/10)*36+lon/10;
/*
        IF THE WORD POSITION HAS CHANGED GET THE NEW ONE DEGREE CELL
        LOGICAL ADDRESS
*/
      if (word!=pword) {
/*
         COMPUTE THE LOGICAL AND PHYSICAL ADDRESS OF THE ONE DEGREE
         CELL POINTER WITHIN THE TEN DEGREE MAP
*/
         lrec = (word*4)/32+1;
         addr = (lrec/(long)(LPERP))*PHYREC;
/*
         IF THE PHYSICAL ADDRESS HAS CHANGED, READ A NEW RECORD
*/
         if (addr!=paddr) {
            lstat = lseek(lunfil,addr,SEEK_SET);
            lstat = read(lunfil,irec10,PHYREC);
            paddr = addr;
         }
/*
         COMPUTE THE BYTE POSITION OF THE ONE DEGREE CELL LOGICAL
         ADDRESS WITHIN THE PHYSICAL RECORD
*/
         pnt = ((word+8)*4)%PHYREC;

/*
         BUILD THE LOGICAL ADDRESS FROM THE 4 BYTES AT 'PNT'
*/
         *rec01 = irec10[pnt]*16777216+irec10[pnt+1]*65536+
         irec10[pnt+2]*256+irec10[pnt+3];
         pword = word;
      }
/*
        IF THE TEN DEGREE SQUARE IS ALL LAND OR WATER, SET THE FLAG
*/
      if (*rec01<3) {
         *tenflg = (int)*rec01%2;
      }
      else {
         *tenflg = -1;
      }
      return ;
```

```
}

/*
        FUNCTION RDONE : READS THE ONE DEGREE CELL MAP.  RETURNS A
        FLAG THAT INDICATES IF THE ENTIRE SQUARE IS EITHER LAND OR
        WATER.  IF NOT, THE VARIABLE 'REC00' IS SET TO THE LOGICAL
        ADDRESS OF THE ONE DEGREE DATA RECORD.  THE FLAG RETURNED IS
        SET TO 1 FOR LAND OR 0 FOR WATER.

             AUTHOR : JAN C. DEPNER, 04/03/89

        ARGUMENTS :

         lat      -  (R) LATITUDE DEGREES OF THE POINT (0 - 180)
         lon      -  (R) LONGITUDE DEGREES OF THE POINT (0 - 360)
         oneflg   -  (C) ALL LAND/WATER FLAG RETURNED

        VARIABLES :

         PHYREC   -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
                    RECORD IN THE DATA BASE.
         celmsk   -  (I) ARRAY CONTAINING THE LOGICAL ADDRESSES OF
                    THE ONE DEGREE CELL DATA RECORDS.
         irec01   -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
                    DATA BASE.
         LPERP    -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
         rec00    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
                   (NOT USED IN THIS ROUTINE).
         rec01    -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
         paddr    -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
*/

void rdone(lunfil,rec01,rec00,lat,lon,oneflg)
int lunfil, lat, lon;
long rec01, *rec00;
char *oneflg;
{

      static long celmsk[10][10], paddr = -1, prec01 = -1;
      static unsigned char irec01[PHYREC];
      static int power[] = {1,2,4,8,16,32,64,128};
      long addr, cell, lstat;
      int sum, bpos, byt, bit, set, i, j, latp, lonp;

/*
        COMPUTE THE CELL POSITION WITHIN THE PHYSICAL RECORD AND THE
        PHYSICAL RECORD ADDRESS
*/
      cell = (rec01%(long)(LPERP))*32;
      addr = (rec01/(long)(LPERP))*PHYREC;
/*
        IF LOGICAL RECORD HAS CHANGED, BUILD A NEW CELL MASK
*/
      if (rec01!=prec01) {
/*
         IF THE ADDRESS HAS CHANGED, READ A NEW RECORD
*/
         if (addr!=paddr) {
            lstat = lseek(lunfil,addr,0);
            lstat = read(lunfil,irec01,PHYREC);
            paddr = addr;
         }
/*
         BUILD THE CELL MASK FROM THE 2 BIT RECORDS WITHIN THE LOGICAL
```

```
           RECORD; 00 - BOTH LAND AND WATER, 01 - LAND, 10 - WATER.  IF
           THE 2 BIT RECORD IS 00, SET THE CELL MASK VALUE TO THE LOGICAL
           ADDRESS OF THE ONE DEGREE DATA RECORD
*/
           sum = 0;
           for (i=0; i<=9; i++) {
              for (j=0; j<=9; j++) {
/*
                 COMPUTE THE BIT POSITION WITHIN THE PHYSICAL RECORD, THE
                 BYTE POSITION WITHIN THE PHYSICAL RECORD, AND THE BIT
                 POSITION WITHIN THE BYTE
*/
               bpos = (i*10+j)*2;
               byt = bpos/8;
               bit = bpos%8;
               set = (irec01[cell+byt]/power[bit])%4;
               if (!set) {
                  sum++;
                  celmsk[j][i] = rec01+sum;
               }
               else {
                  celmsk[j][i] = set;
               }
              }
           }
           prec01 = rec01;
        }
/*
          GET THE ADDRESS, OR GET THE FLAG IF ALL LAND OR WATER
*/
        lonp = lon%10;
        latp = lat%10;
        if (celmsk[lonp][latp]>2) {
           *oneflg = -1;
           *rec00 = celmsk[lonp][latp];
        }
        else {
           *oneflg = celmsk[lonp][latp]%2;
        }
        return;
}

/*
        FUNCTION ALLMSK : SET THE ENTIRE MASK TO EITHER 'L' OR 'W'.

             AUTHOR : JAN C. DEPNER, 04/03/89

        ARGUMENTS :

         mask      -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
                      MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
                      THE LATITUDE AND LONGITUDE POINT REQUESTED
         alflag    -  (C) ALL LAND/WATER FLAG
*/

void allmsk(alflag,mask)
char alflag, mask[];
{

        int i, j;

        for (i=0; i<=59; i++) {
           for (j=0; j<=59; j++) {
              mask[i*60+j] = alflag;
```

```
        }
      }
      return;
}

/*
      FUNCTION BLDMSK : BUILDS THE MASK FROM THE ONE DEGREE DATA
      RECORDS IN THE DATA BASE.

           AUTHOR : JAN C. DEPNER, 04/03/89

      ARGUMENTS :

       mask      -  (C) ARRAY CONTAINING THE ONE MINUTE GRID OF LAND
                    MASK POINTS FOR THE ONE DEGREE CELL THAT CONTAINS
                    THE LATITUDE AND LONGITUDE POINT REQUESTED

      VARIABLES :

       PHYREC    -  (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
                    RECORD IN THE DATA BASE.
       irec00    -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
                    DATA BASE.
       LPERP     -  (I) LOGICAL RECORDS PER PHYSICAL RECORD.
       rec00     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
                    (NOT USED IN THIS ROUTINE).
       rec01     -  (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
       paddr     -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.
*/

void bldmsk(lunfil,rec00,rec01,mask)
int lunfil;
long rec00, rec01;
char mask[];
{

      static long paddr = -1;
      static unsigned char irec00[PHYREC];
      int start, type, count, finish, j;
      long addr, pos, lstat;
      unsigned char mbyte, dflag;

      void movpos();

      start = 0;
/*
        COMPUTE THE FIRST BYTE POSITION WITHIN THE PHYSICAL RECORD, AND THE
        PHYSICAL RECORD ADDRESS
*/
      pos = (rec00%(long)(LPERP))*32;
      addr = (rec00/(long)(LPERP))*PHYREC;
/*
        IF THE ADDRESS HAS CHANGED, READ A NEW RECORD
*/
      if (addr!=paddr) {
printf ("paddr = %d %o, addr = %d %o\n",paddr,paddr,addr,addr);
          lstat = lseek(lunfil,addr,0);
          lstat = read(lunfil,irec00,PHYREC);
          paddr = addr;
      }
/*
        MAIN LOOP, GETS TYPE AND COUNT FROM BYTES.  FIRST TWO BITS ARE TYPE
        FLAGS; 00 - 1 BYTE WATER, 01 - 1 BYTE LAND, 10 - 2 BYTE WATER,
        11 - 2 BYTE LAND.  NEXT 6 OR 14 BITS ARE COUNT
```

```
*/
      mbyte = irec00[pos];
      while (mbyte) {
/*
         IF THIS IS A TWO BYTE RECORD, SET DFLAG
*/
         if (mbyte>=128) {
            mbyte = mbyte-128;
            dflag = 1;
         }
         else {
            dflag = 0;
         }
/*
         GET THE TYPE AND COUNT
*/
         type = mbyte/64;
         count = mbyte-type*64;
/*
         IF THIS IS A TWO BYTE RECORD, GET THE REST OF THE COUNT FROM
         THE NEXT BYTE
*/
         if (dflag) {
            movpos(lunfil,&pos,irec00,&paddr,rec01);
            count = count*256+irec00[pos];
         }
/*
         LOOP THROUGH THE MASK AND SET THE CHARACTERS
*/
         finish = start+count-1;
         for (j=start; j<=finish; j++) {
            mask[j] = type;
         }
/*
         MOVE THE BYTE POSITION AND GO TO TOP OF LOOP
*/
         movpos(lunfil,&pos,irec00,&paddr,rec01);
         start = finish+1;
         mbyte = irec00[pos];
      }
/*
        FLIP THE TYPE FLAG AND SET THE REMAINDER OF THE MASK
*/
      type = 1-type;
      for (j=start; j<=3599; j++) {
         mask[j] = type;
      }
      return;
}

/*
      FUNCTION MOVPOS : INCREMENTS THE BYTE POSITION AND CHANGES
      TO THE NEXT OVERFLOW RECORD IN THE CHAIN IF NECCESSARY.

           AUTHOR : JAN C. DEPNER, 04/03/89

      ARGUMENTS :

       pos       -  (I) BYTE POSITION WITHIN THE PHYSICAL RECORD.
       irec00    -  (C) ARRAY CONTAINING AN INPUT RECORD FROM THE
                    DATA BASE.
       paddr     -  (I) PREVIOUS PHYSICAL RECORD ADDRESS.

      VARIABLES :
```

```
        PHYREC    - (I) CONSTANT SET TO NUMBER OF BYTES IN A PHYSICAL
                    RECORD IN THE DATA BASE.
        LPERP     - (I) LOGICAL RECORDS PER PHYSICAL RECORD.
        rec00     - (I) LOGICAL ADDRESS OF THE ONE DEGREE DATA RECORD
                    (NOT USED IN THIS ROUTINE).
        rec01     - (I) LOGICAL ADDRESS OF THE ONE DEGREE CELL MAP.
*/

void movpos(lunfil,pos,irec00,paddr,rec01)
int lunfil;
unsigned char irec00[];
long *pos, *paddr, rec01;
{

      long lrec, addr, lstat;

      (*pos)++;
/*
        IF THIS IS THE END OF THE LOGICAL RECORD, GET THE OVERFLOW POINTER
        FROM THE LAST 2 BYTES OF THE RECORD
*/
      if ((*pos+2)%32==0) {
         lrec = irec00[*pos]*256+irec00[*pos+1]+rec01;
/*
         IF LREC = REC01, THERE IS NO OVERFLOW POINTER AND THIS IS THE
         END OF THE CHAIN
*/
         if (lrec!=rec01) {
/*
            COMPUTE THE PHYSICAL ADDRESS, IF IT HAS CHANGED, READ A NEW
            RECORD
*/
            addr = (lrec/(long)(LPERP))*PHYREC;
            if (addr!=*paddr) {
             lstat = lseek(lunfil,addr,0);
             lstat = read(lunfil,irec00,PHYREC);
             *paddr = addr;
            }
/*
            COMPUTE THE NEW BYTE POSITION WITHIN THE PHYSICAL RECORD
*/
            *pos = (lrec%(long)(LPERP))*32;
         }
      }
      return;
}
```